

EXHIBIT 4

Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

**Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)**

Communication Interface Technologies, LLC (“CIT”) provides this preliminary and exemplary infringement analysis with respect to infringement of U.S. Patent No. 6,574,239, entitled “*Virtual Connection of a Remote Unit to a Server*” (“the ’239 patent”) by 7-Eleven, Inc. (“7-Eleven”). The following chart illustrates an exemplary analysis regarding infringement by 7-Eleven’s commercial mobile device application(s) including the 7-Eleven, Inc. mobile app, 7NOW: Food & Alcohol Delivery mobile app, 7-Eleven Cashierless mobile app, and the 7-Track mobile app, along with any hardware and/or software for provisioning that mobile device application (collectively, the “Accused Instrumentalities”). Upon information and belief, the exemplary version herein and previous versions of the Accused Instrumentalities distributed prior to expiration of the patents-in-suit operated materially in the same manner.

The analysis set forth below is based only upon information from publicly available resources regarding the Accused Instrumentalities, as 7-Eleven has not yet provided any non-public information.

Unless otherwise noted, CIT contends that 7-Eleven directly infringes the ’239 patent in violation of 35 U.S.C. § 271(a) by selling, offering to sell, making, using, and/or importing the Accused Instrumentalities. The following exemplary analysis demonstrates that infringement.

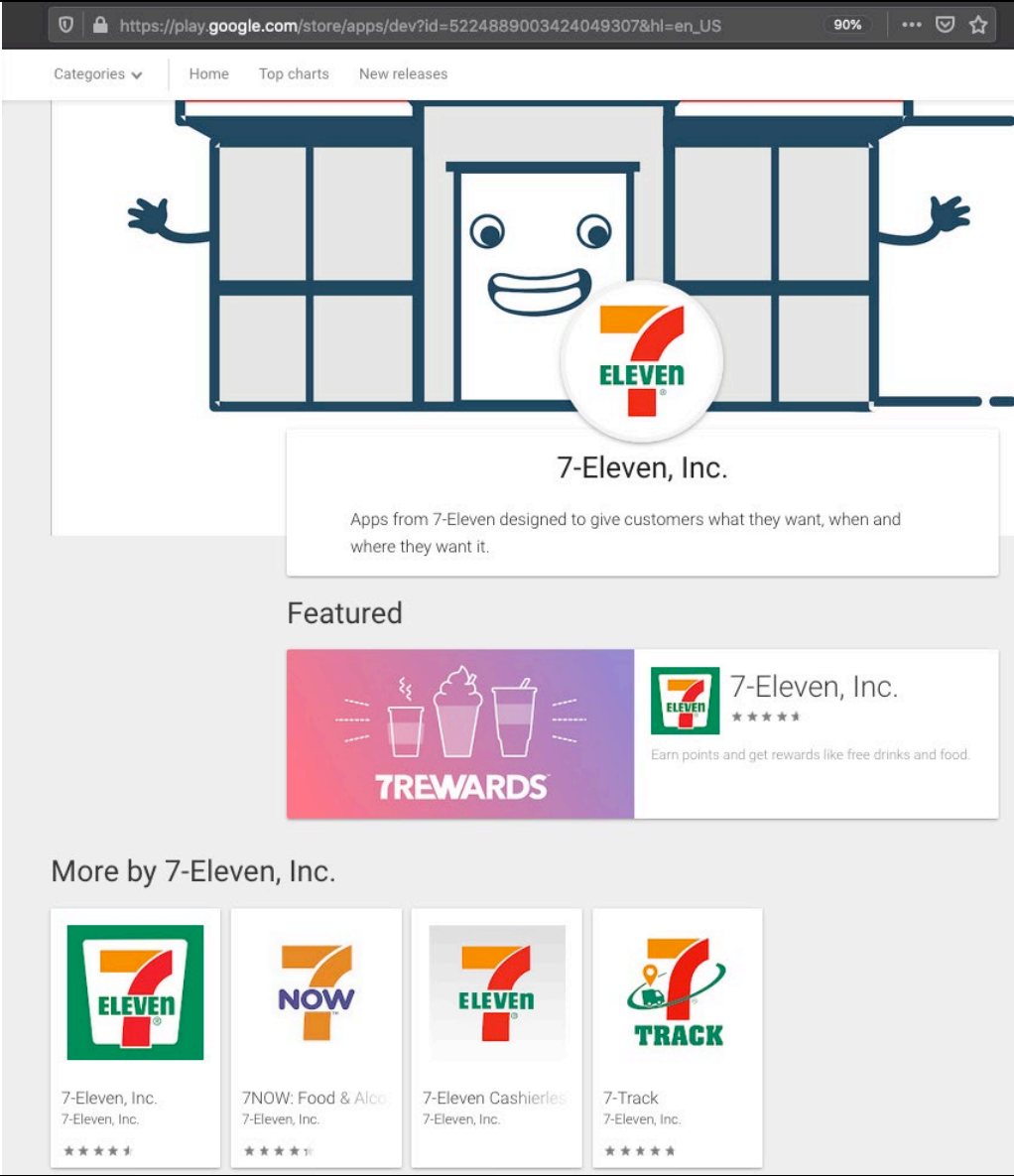
Unless otherwise noted, CIT believes and contends that each element of each claim asserted herein is literally met through 7-Eleven’s provision of the Accused Instrumentalities. However, to the extent that 7-Eleven attempts to allege that any asserted claim element is not literally met, CIT believes and contends that such elements are met under the doctrine of equivalents. More specifically, in its investigation and analysis of the Accused Instrumentalities, CIT did not identify any substantial differences between the elements of the patent claims and the corresponding features of the Accused Instrumentalities, as set forth herein. In each instance, the identified feature of the Accused Instrumentalities performs at least substantially the same function in substantially the same way to achieve substantially the same result as the corresponding claim element.

CIT notes that the present claim chart and analysis are necessarily preliminary in that CIT has not obtained substantial discovery from 7-Eleven nor has 7-Eleven disclosed any detailed analysis for its non-infringement position, if any. Further, CIT does not have the benefit of claim construction or expert discovery. CIT reserves the right to supplement and/or amend the positions taken in this preliminary and exemplary infringement analysis, including with respect to literal infringement and infringement under the doctrine of equivalents, if and when warranted by further information obtained by CIT, including but not limited to information adduced through information exchanges between the parties, fact discovery, claim construction, expert discovery, and/or further analysis.

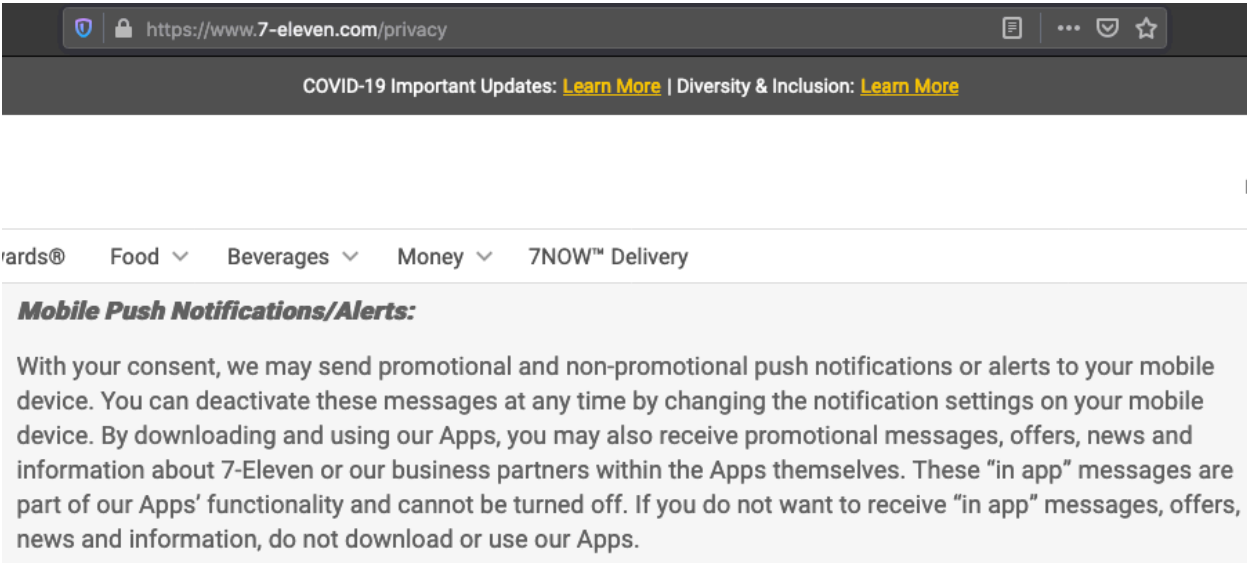
Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

	Claim 7	7-Eleven Service
7	For use in controlling a virtual session on a server, a method comprising:	<p>A method is specified for controlling a virtual session on a server. The definition of a virtual session is described section 7a below.</p> <p><i>See https://play.google.com/store/apps/dev?id=5224889003424049307</i></p>

Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

		
7a	establishing a virtual session with a remote unit, the virtual session being instantiated to support at least one	Wireless push notification messages are sent over Transport Layer Security (TLS) sessions. Each Push message includes an encrypted push token as per Endnote #1. The push token is sent in a Push Notification message over TLS sessions from the 7-Eleven Server backend to the 7-Eleven

Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

<p>application layer program;</p>	<p>App (application program, application layer program) running on a user's smartphone or tablet (remote unit).</p> <p>In the 7-Eleven application, for example, a push notification contains information related to 7-Eleven goods and/or services.</p> <p>See https://www.7-eleven.com/privacy</p>  <p>Also, the Server Application and the client side application establish a separate TLS connection for traditional client-server communications. For example, the 7-Eleven Application Server program establishes a TLS session with the 7-Eleven App.</p> <p>The TLS session used for client-server communications between the 7-Eleven Application Server and the 7-Eleven App correspond to the recited virtual session.</p> <p>TLS session use a full handshake sequence that is used to establish connection parameters, and an abbreviated handshake sequence that is used to resume the TLS session from an inactive or dormant state to an active state whereby new payload data can be sent via the virtual session once again.</p>
-----------------------------------	---

Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

		<p><i>See</i> Endnote #2 for a discussion of the virtual session aspects of TLS.</p> <p>Mobile applications communicate with their application server via TLS connections. These TLS connections are established at the time the app is installed or launched and can be resumed at a later time using a session token. <i>See</i> Endnote #2 for a discussion of TLS.</p> <p><i>See</i> https://threema.ch/press-files/cryptography_whitepaper.pdf - Android uses TLS 1.2 resumable sessions.</p> <p><i>See</i> https://www.icir.org/johanna/papers/conext17android.pdf - Android also supports TLS and secure connections between client app and server are ubiquitously used.</p> <p><i>See</i> https://developer.android.com/training/articles/security-ssl – “The Secure Sockets Layer (SSL)—now technically known as Transport Layer Security (TLS)—is a common building block for encrypted communications between clients and servers.” Note that certain certificates are used, and these are used to help create Android Device Tokens used in Push Notification messages. <i>See</i> Endnotes #1, and #2.</p>
7b	placing the virtual session in an inactive state;	<i>See</i> Endnote #2. Note when the application data phase is finished, the TLS client-server data session is placed back into the inactive state. Hence the end of application data marker is the signal used to place the virtual session into the inactive state.
7c	sending a signal indicative of an incoming communication request and an application-program identifying packet to said remote unit, said application-program identifying packet identifying an application program that needs to resume a virtual session and communicate with said remote unit; and	<p>7-Eleven Application server causes a push notification message (incoming communication) to be sent to the 7-Eleven App running on the user’s smartphone or tablet device (remote unit). <i>See</i> Endnote #1.</p> <p>In the 7-Eleven App, for example, a push notification contains information related to 7-Eleven goods and/or services.</p> <p><i>See</i> https://www.7-eleven.com/privacy</p>

Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

		<p>The signal indicative of an incoming communication request will request the application program running on the remote unit to resume the TLS session used for client-server communications between the 7-Eleven Application server and the 7-Eleven App running on the user’s smartphone or tablet device (remote unit). The application-program identifying packet will include information used to identify the App) on the remote unit.</p> <p>The 7-Eleven server and the 7-Eleven App will resume a TLS session so that the server and the remote unit can resume communications. To do so the 7-Eleven App will invoke a protocol stack within the remote unit to communicate back to the server via the remote unit.</p> <p>See Endnote #1 for a discussion of how each new set of data payloads coming into the 7-Eleven application includes an app-specific device token. The app-specific device token is indicative of the 7-Eleven application running on the remote unit. Each incoming wireless push notification message contains the app-specific device token which is part of the application-program identifying packet.</p>
7d	placing the virtual session back into the active state and transferring data	Based upon user selection of information associated with the above-mentioned push notification, the Transport Layer Security (TLS) session between the 7-Eleven Application Server and the 7-

Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

<p>between the application and the remote unit via the virtual session in response to said step of sending.</p>	<p>Eleven App is resumed as discussed in Endnote #2. TLS session resumption means that the TLS session is placed back into the active state using an abbreviated handshake sequence so that new application data can be passed between the 7-Eleven Application Server and the App running on the user's smartphone or tablet.</p> <p><i>See Endnote#2 for a discussion of TLS session resumption. See also, https://docs.microsoft.com/en-us/windows/desktop/secauthn/tls-handshake-protocol.</i></p>
---	---

Endnote#1 - App-Specific Device Token

<https://help.pushwoosh.com/hc/en-us/articles/360000364923-What-is-a-Device-token->

Question:

What is a Device token?

Answer:

Push token (device token) - is a unique key for the app-device combination which is issued by the Apple or Google push notification gateways. It allows gateways and push notification providers to route messages and ensure the notification is delivered only to the unique app-device combination for which it is intended.

iOS device push tokens are strings with 64 hexadecimal symbols. Push token example:

03df25c845d460bcdad7802d2vf6fc1dfde97283bf75cc993eb6dca835ea2e2f

Make sure that iOS push tokens you use when targeting specific devices in your API requests are in lower case.

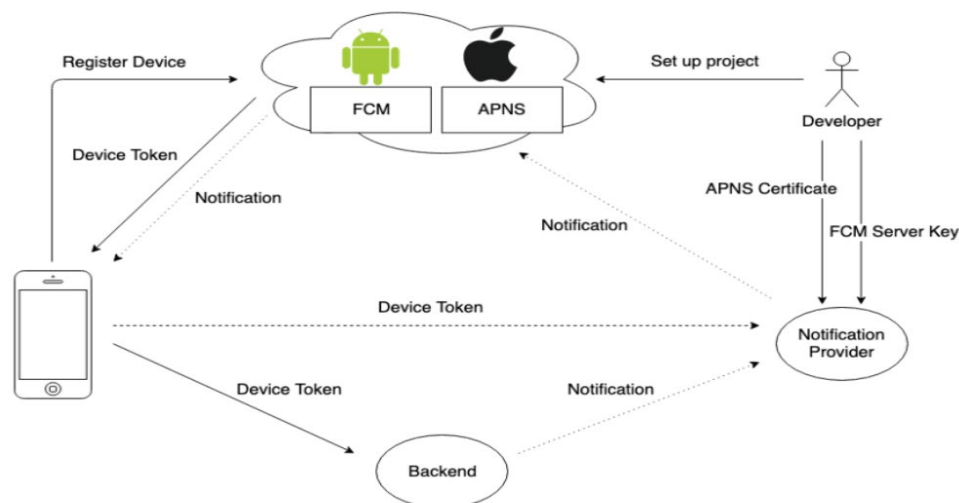
Android device push tokens can differ in length (usually below 255 characters), and usually start with APA...Push token example:

APA91bFoi3lMMre9G3XzR1LrF4ZT82_15MsMdEICogXSLB8-MrdkRuRQFwNI5u8Dh0cI90ABD3BOKnxkEla8cGdisbDH15cVIkZah5QUhSAxxz4Roa7b4xy9tvx9iNSYw-eXBYyd8k1XKf8Q_Qq1X9-x-U-Y79vdPq

Note: The Android device push tokens correspond to the app-specific device token terminology used in the claim charts.

<https://dev.to/jakubkoci/react-native-push-notifications-313i>

Architecture



Note: In the above Architecture, the “backend” corresponds to the backend of the Application Server. The Device token corresponds to a specific App running on a specific device. That is what is meant by the app-specific device token in the claim charts.

<https://firebase.google.com/docs/cloud-messaging/android/first-message>

Access the registration token

To send a message to a specific device, you need to know that device's registration token. Because you'll need to enter the token in a field in the Notifications console to complete this tutorial, make sure to copy the token or securely store it after you retrieve it.

Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

On initial startup of your app, the FCM SDK generates a registration token for the client app instance. If you want to target single devices or create device groups, you'll need to access this token by extending `FirebaseMessagingService` and overriding `onNewToken`.

This section describes how to retrieve the token and how to monitor changes to the token. Because the token could be rotated after initial startup, you are strongly recommended to retrieve the latest updated registration token.

The registration token may change when:

- The app deletes Instance ID
- The app is restored on a new device
- The user uninstalls/reinstall the app
- The user clears app data.”

<https://firebase.google.com/docs/cloud-messaging/concept-options>

For example, here is a JSON-formatted notification message in an IM app. The user can expect to see a message with the title "Portugal vs. Denmark" and the text "great match!" on the device:

```
{
  "message": {
    "token": "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...", ←-- App-specific token
    "notification": {
      "title": "Portugal vs. Denmark",
      "body": "great match!"
    }
  }
}
```

<https://firebase.google.com/docs/cloud-messaging/android/client>

Retrieve the current registration token

When you need to retrieve the current token, call `FirebaseInstanceId.getInstance().getInstanceId()`:

```

FirebaseInstanceId.getInstance().getInstanceId()
    .addOnCompleteListener(new OnCompleteListener<InstanceIdResult>() {
        @Override
        public void onComplete(@NonNull Task<InstanceIdResult> task) {
            if (!task.isSuccessful()) {
                Log.w(TAG, "getInstanceId failed", task.getException());
                return;
            }

            // Get new Instance ID token
            String token = task.getResult().getToken();

            // Log and toast
            String msg = getString(R.string.msg_token_fmt, token);
            Log.d(TAG, msg);
            Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT).show();
        }
    });

```

MainActivity.java

Monitor token generation

The `onNewToken` callback fires whenever a new token is generated.

```

/**
 * Called if InstanceID token is updated. This may occur if the security of

```

```

* the previous token had been compromised. Note that this is called when the InstanceID token
* is initially generated so this is where you would retrieve the token.
*/
@Override
public void onNewToken(String token) {
    Log.d(TAG, "Refreshed token: " + token);

    // If you want to send messages to this application instance or
    // manage this apps subscriptions on the server side, send the
    // Instance ID token to your app server.
    sendRegistrationToServer(token);
}

```

After you've obtained the token, you can send it to your app server and store it using your preferred method. See the Instance ID API reference [<https://firebase.google.com/docs/reference/android/com/google/firebase/iid/FirebaseInstanceId>] for full detail on the API.

Endnote#2 - Transport Layer Security and Virtual Sessions

Transport Layer Security (TLS) is used by both Apple iOS and Android based devices. The handshake diagrams in this endnote use Apple iOS as an example but apply equally to Android type implementations.

<https://developer.android.com/training/articles/security-ssl>

“The Secure Sockets Layer (SSL)—now technically known as Transport Layer Security (TLS)—is a common building block for encrypted communications between clients and servers.”

<https://android-developers.googleblog.com/2018/04/protecting-users-with-tls-by-default-in.html>

“Android is committed to keeping users, their devices, and their data safe. One of the ways that we keep data safe is by protecting all data that enters or leaves an Android device with Transport Layer Security (TLS) in transit.”

Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

A. Razaghpanah et al., *Studying TLS Usage in Android Apps*, CoNEXT '17, Dec.12-15, 2017, Incheon, Republic of Korea
http://abbas.rpanah.ir/publications/conext2017_tls_paper.pdf

“A History of TLS Support in Android: Android has supported TLS 1.0 since its first version released in 2008 and TLS 1.1 and TLS 1.2 since 2012.”

“However, other protocols such as secure email (42 apps) and Google’s Cloud Messaging service for push notifications (9 apps) [11, 47] also use TLS.”

<https://developer.ibm.com/customer-engagement/docs/watson-marketing/ibm-engage-2/tls-1-2-migration-for-mobile-push-clients/>

What will happen on devices that are unable to support TLS 1.2?

Devices which do not support TLS 1.2 will be unable to connect to our WCA servers. This will prevent users of those devices from:

- Registering new mobile user IDs
- Updating push tokens
- Receiving inbox messages
- Receiving In-app messages

Note: As the above link shows, the creation of the App IDs of Endnote #1 are linked to the TLS protocol being run on the TLS-enabled Push-Notification channel.

<https://tools.ietf.org/html/rfc5246>

F.1.4. Resuming Sessions

When a connection is established by resuming a session, new ClientHello.random and ServerHello.random values are hashed with the session's master_secret. Provided that the master_secret has not been compromised and that the secure hash operations used to produce the encryption keys and MAC keys are secure, the connection should be secure and effectively independent from previous connections. Attackers cannot use known encryption keys or MAC secrets to compromise the master_secret without breaking the secure hash operations.

Sessions cannot be resumed unless both the client and server agree. If either party suspects that the session may have been compromised, or that certificates may have expired or been revoked, it should force a full handshake. An upper limit of 24 hours is suggested for session ID lifetimes,

Analysis of Infringement of U.S. Patent No. 6,574,239 by 7-Eleven, Inc.
(Based on Public Information Only)

since an attacker who obtains a master_secret may be able to impersonate the compromised party until the corresponding session ID is retired. Applications that may be run in relatively insecure environments should not write session IDs to stable storage.

<https://tools.ietf.org/html/rfc5077>

Abstract

This document describes a mechanism that enables the Transport Layer Security (TLS) server to resume sessions and avoid keeping per-client session state. The TLS server encapsulates the session state into a ticket and forwards it to the client. The client can subsequently resume a session using the obtained ticket.

3. Protocol

This specification describes a mechanism to distribute encrypted session-state information in the form of a ticket. The ticket is created by a TLS server and sent to a TLS client. The TLS client presents the ticket to the TLS server to resume a session.